Ejemplo de como obtener los angulos de orientacion en android

Ahora usa dos sensores (ACCELEROMETER y MAGNETIC\_FIELD) para obtener esa información.

public class CompassActivity extends Activity implements SensorEventListener {

private SensorManager mSensorManager;

Sensor accelerometer;

Sensor magnetometer;

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(mCustomDrawableView); // Register the sensor listeners

mSensorManager = (SensorManager)getSystemService(SENSOR\_SERVICE);

accelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE\_ACCELEROMETER);

magnetometer = mSensorManager.getDefaultSensor(Sensor.TYPE\_MAGNETIC\_FIELD);

}

protected void onResume() {

super.onResume();

mSensorManager.registerListener(this, accelerometer, SensorManager.SENSOR\_DELAY\_UI);

mSensorManager.registerListener(this, magnetometer, SensorManager.SENSOR\_DELAY\_UI);

}

protected void onPause() {

super.onPause();

mSensorManager.unregisterListener(this);

}

public void onAccuracyChanged(Sensor sensor, int accuracy) { }

float[] mGravity;

float[] mGeomagnetic;

public void onSensorChanged(SensorEvent event) {

if (event.sensor.getType() == Sensor.TYPE\_ACCELEROMETER)

mGravity = event.values;

if (event.sensor.getType() == Sensor.TYPE\_MAGNETIC\_FIELD)

mGeomagnetic = event.values;

if (mGravity != null && mGeomagnetic != null) {

float R[] = new float[9];

float I[] = new float[9];

boolean success = SensorManager.getRotationMatrix(R, I, mGravity, mGeomagnetic);

if (success) {

float orientation[] = new float[3];

SensorManager.getOrientation(R, orientation);

azimut = orientation[0]; // orientation contains: azimut, pitch and roll

}

}

}

}

Permissions:

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"/>

Hay 2 formas de obtener los 3 valores de rotación (acimut, paso, balanceo).  
  
Una es registrar un oyente de un tipo TYPE\_ORIENTATION. Es la forma más fácil y obtengo un rango correcto de valores de cada rotación, como dice la documentación: azimuth: [0, 359] tono: [-180, 180] roll: [-90, 90]  
  
El otro, el más preciso y complejo de entender la primera vez que lo ves. Android lo recomienda, así que quiero usarlo, pero obtengo valores diferentes.  
  
acimut: [-180, 180]. -180/180 es S, 0 i N, 90 E y -90 W.  
tono: [-90, 90]. 90 es 90, -90 es -90, 0 es 0 pero -180/180 (acostado con la pantalla hacia abajo) es 0.  
rollo: [-180, 180].  
  
Debería obtener los mismos valores pero con decimales, ¿verdad?  
  
Tengo el siguiente código:

aValues = new float[3];

mValues = new float[3];

sensorListener = new SensorEventListener (){

public void onSensorChanged (SensorEvent event){

switch (event.sensor.getType ()){

case Sensor.TYPE\_ACCELEROMETER:

aValues = event.values.clone ();

break;

case Sensor.TYPE\_MAGNETIC\_FIELD:

mValues = event.values.clone ();

break;

}

float[] R = new float[16];

float[] orientationValues = new float[3];

SensorManager.getRotationMatrix (R, null, aValues, mValues);

SensorManager.getOrientation (R, orientationValues);

orientationValues[0] = (float)Math.toDegrees (orientationValues[0]);

orientationValues[1] = (float)Math.toDegrees (orientationValues[1]);

orientationValues[2] = (float)Math.toDegrees (orientationValues[2]);

azimuthText.setText ("azimuth: " + orientationValues[0]);

pitchText.setText ("pitch: " + orientationValues[1]);

rollText.setText ("roll: " + orientationValues[2]);

}

public void onAccuracyChanged (Sensor sensor, int accuracy){}

};

Por favor ayuda. Es muy frustrante  
  
¿Tengo que tratar con esos valores o estoy haciendo algo mal?  
  
Gracias.

Sé que estoy jugando al nigromante de hilos aquí, pero he estado trabajando mucho en esto últimamente, así que pensé en lanzar mi 2 ¢.  
  
El dispositivo no contiene brújula ni inclinómetros, por lo que no mide acimut, cabeceo ni balanceo directamente. (Llamamos a esos ángulos de Euler, por cierto). En cambio, usa acelerómetros y magnetómetros, que producen vectores XYZ de 3 espacios. Estos se utilizan para calcular los valores de acimut, etc.  
  
Los vectores están en el espacio de coordenadas del dispositivo:  
  
  
Las coordenadas mundiales tienen Y mirando hacia el norte, X mirando hacia el este y Z hacia arriba:  
  
Coordenadas mundiales  
  
Por lo tanto, la orientación "neutral" de un dispositivo está tendida de espaldas sobre una mesa, con la parte superior del dispositivo orientada hacia el norte.  
  
El acelerómetro produce un vector en la dirección "ARRIBA". El magnetómetro produce un vector en la dirección "norte". (Tenga en cuenta que en el hemisferio norte, esto tiende a apuntar hacia abajo debido a la inmersión magnética).  
  
El vector del acelerómetro y el vector del magnetómetro se pueden combinar matemáticamente a través de SensorManager.getRotationMatrix () que devuelve una matriz de 3x3 que asignará los vectores en las coordenadas del dispositivo a las coordenadas mundiales o viceversa. Para un dispositivo en la posición neutral, esta función devolvería la matriz de identidad.  
  
Esta matriz no varía con la orientación de la pantalla. Esto significa que su aplicación necesita conocer la orientación y compensar en consecuencia.  
  
SensorManager.getOrientation () toma la matriz de transformación y calcula los valores azimuth, pitch y roll. Estos se toman en relación con un dispositivo en la posición neutral.  
  
No tengo idea de cuál es la diferencia entre llamar a esta función y simplemente usar el sensor TYPE\_ORIENTATION, excepto que la función te permite manipular la matriz primero.  
  
Si el dispositivo está inclinado hacia arriba a 90 ° o cerca de él, entonces el uso de ángulos de Euler se desmorona. Este es un caso degenerado matemáticamente. En este ámbito, ¿cómo se supone que el dispositivo debe saber si está cambiando el acimut o el balanceo?  
  
La función SensorManager.remapCoordinateSystem () se puede utilizar para manipular la matriz de transformación para compensar lo que puede saber sobre la orientación del dispositivo. Sin embargo, mis experimentos han demostrado que esto no cubre todos los casos, ni siquiera algunos de los más comunes. Por ejemplo, si quiere remapear para un dispositivo en posición vertical (por ejemplo, para tomar una foto), querrá multiplicar la matriz de transformación por esta matriz:  
  
1 0 0  
0 0 1  
0 1 0  
  
antes de llamar a getOrientation (), y esta no es una de las reasignaciones de orientación que remapCoordinateSystem () admite [alguien por favor corrígeme si me he perdido algo aquí].  
  
OK, así que todo esto ha sido una manera larga de decir que si estás usando orientación, ya sea desde el sensor TYPE\_ORIENTATION o desde getOrientation (), probablemente lo estás haciendo mal. La única vez que realmente desea los ángulos de Euler es mostrar la información de orientación en una forma fácil de usar, anotar una fotografía, conducir la pantalla del instrumento de vuelo o algo similar.  
  
Si desea realizar cálculos relacionados con la orientación del dispositivo, es casi seguro que es mejor utilizar la matriz de transformación y trabajar con vectores XYZ.  
  
Trabajando como consultor, cada vez que alguien acude a mí con un problema relacionado con los ángulos de Euler, hago una copia de seguridad y les pregunto qué están tratando de hacer realmente, y luego encuentro una manera de hacerlo con vectores en su lugar.  
  
Mirando hacia atrás a su pregunta original, getOrientation () debería devolver tres valores en [-180 180] [-90 90] y [-180 180] (después de convertir de radianes). En la práctica, pensamos en acimut como números en [0 360], por lo que simplemente debe agregar 360 a los números negativos que reciba. Tu código se ve correcto como está escrito. Ayudaría si supiera exactamente qué resultados esperaba y qué obtendría en su lugar.  
  
Editado para agregar: un par de pensamientos más. Las versiones modernas de Android usan algo llamado "fusión de sensores", lo que básicamente significa que todas las entradas disponibles (acelerador, magnetómetro, giroscopio) se combinan en una caja negra matemática (normalmente un filtro de Kalman, pero depende del proveedor). Todos los diferentes sensores (aceleración, campo magnético, giroscopios, gravedad, aceleración lineal y orientación) se toman como salidas de esta caja negra.  
  
Siempre que sea posible, debe usar TYPE\_GRAVITY en lugar de TYPE\_ACCELEROMETER como la entrada a getRotationMatrix ().

Podría estar disparando en la oscuridad aquí, pero si entiendo tu pregunta correctamente, te preguntas por qué obtienes [-179..179] en lugar de [0..360]?  
  
Tenga en cuenta que -180 es lo mismo que +180 y lo mismo que 180 + N \* 360, donde N es un número entero (entero).  
  
En otras palabras, si desea obtener los mismos números que con el sensor de orientación, puede hacer esto:

// x = orientationValues[0];

// y = orientationValues[1];

// z = orientationValues[2];

x = (x + 360.0) % 360.0;

y = (y + 360.0) % 360.0;

z = (z + 360.0) % 360.0;

Esto le dará los valores en el rango [0..360] que desee.

Cuando utiliza el sensor TYPE\_MAGNETOMETER, obtiene los valores X, Y, Z de la intensidad del campo magnético en relación con la orientación del dispositivo. Lo que quiero obtener es convertir estos valores en un marco de referencia global, aclarando: el usuario toma el dispositivo, mide estos valores, que gira el dispositivo en algunos grados alrededor de cualquier eje y obtiene ~ los mismos valores. Por favor, encuentre preguntas similares a continuación: Obtener valores de campo magnético en coordenadas globales ¿Cómo puedo obtener el vector de campo magnético, independientemente de la rotación del dispositivo? En esta solución de muestra de respuesta se describe (es para la aceleración lineal, pero creo que no importa): https://stackoverflow.com/a/11614404/2152255 Lo usé y obtuve 3 valores, X siempre es muy pequeño (no creo que sea correcto), Y y Z están bien, pero aún cambian un poco cuando giro el dispositivo. ¿Cómo podría ajustarse? ¿Y podría resolverse todo? Utilizo un filtro de Kalman simple para aproximar los valores de medición, porque sin él obtengo valores diferentes, incluso si el dispositivo no se mueve / gira en absoluto. Por favor, encuentra mi código a continuación:

import android.app.Activity;

import android.hardware.Sensor;

import android.hardware.SensorEvent;

import android.hardware.SensorEventListener;

import android.hardware.SensorManager;

import android.opengl.Matrix;

import android.os.Bundle;

import android.view.View;

import android.widget.CheckBox;

import android.widget.TextView;

import com.test.statistics.filter.kalman.KalmanState;

import com.example.R;

/\*\*

\* Activity for gathering magnetic field statistics.

\*/

public class MagneticFieldStatisticsGatheringActivity extends Activity implements SensorEventListener {

public static final int KALMAN\_STATE\_MAX\_SIZE = 80;

public static final double MEASUREMENT\_NOISE = 5;

/\*\* Sensor manager. \*/

private SensorManager mSensorManager;

/\*\* Magnetometer spec. \*/

private TextView vendor;

private TextView resolution;

private TextView maximumRange;

/\*\* Magnetic field coordinates measurements. \*/

private TextView magneticXTextView;

private TextView magneticYTextView;

private TextView magneticZTextView;

/\*\* Sensors. \*/

private Sensor mAccelerometer;

private Sensor mGeomagnetic;

private float[] accelerometerValues;

private float[] geomagneticValues;

/\*\* Flags. \*/

private boolean specDefined = false;

private boolean kalmanFiletring = false;

/\*\* Rates. \*/

private float nanoTtoGRate = 0.00001f;

private final int gToCountRate = 1000000;

/\*\* Kalman vars. \*/

private KalmanState previousKalmanStateX;

private KalmanState previousKalmanStateY;

private KalmanState previousKalmanStateZ;

private int previousKalmanStateCounter = 0;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main2);

mSensorManager = (SensorManager) getSystemService(SENSOR\_SERVICE);

mAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE\_ACCELEROMETER);

mGeomagnetic = mSensorManager.getDefaultSensor(Sensor.TYPE\_MAGNETIC\_FIELD);

vendor = (TextView) findViewById(R.id.vendor);

resolution = (TextView) findViewById(R.id.resolution);

maximumRange = (TextView) findViewById(R.id.maximumRange);

magneticXTextView = (TextView) findViewById(R.id.magneticX);

magneticYTextView = (TextView) findViewById(R.id.magneticY);

magneticZTextView = (TextView) findViewById(R.id.magneticZ);

mSensorManager.registerListener(this, mAccelerometer, SensorManager.SENSOR\_DELAY\_FASTEST);

mSensorManager.registerListener(this, mGeomagnetic, SensorManager.SENSOR\_DELAY\_FASTEST);

}

/\*\*

\* Refresh statistics.

\*

\* @param view - refresh button view.

\*/

public void onClickRefreshMagneticButton(View view) {

resetKalmanFilter();

}

/\*\*

\* Switch Kalman filtering on/off

\*

\* @param view - Klaman filetring switcher (checkbox)

\*/

public void onClickKalmanFilteringCheckBox(View view) {

CheckBox kalmanFiltering = (CheckBox) view;

this.kalmanFiletring = kalmanFiltering.isChecked();

}

@Override

public void onSensorChanged(SensorEvent sensorEvent) {

if (sensorEvent.accuracy == SensorManager.SENSOR\_STATUS\_UNRELIABLE) {

return;

}

synchronized (this) {

switch(sensorEvent.sensor.getType()){

case Sensor.TYPE\_ACCELEROMETER:

accelerometerValues = sensorEvent.values.clone();

break;

case Sensor.TYPE\_MAGNETIC\_FIELD:

if (!specDefined) {

vendor.setText("Vendor: " + sensorEvent.sensor.getVendor() + " " + sensorEvent.sensor.getName());

float resolutionValue = sensorEvent.sensor.getResolution() \* nanoTtoGRate;

resolution.setText("Resolution: " + resolutionValue);

float maximumRangeValue = sensorEvent.sensor.getMaximumRange() \* nanoTtoGRate;

maximumRange.setText("Maximum range: " + maximumRangeValue);

}

geomagneticValues = sensorEvent.values.clone();

break;

}

if (accelerometerValues != null && geomagneticValues != null) {

float[] Rs = new float[16];

float[] I = new float[16];

if (SensorManager.getRotationMatrix(Rs, I, accelerometerValues, geomagneticValues)) {

float[] RsInv = new float[16];

Matrix.invertM(RsInv, 0, Rs, 0);

float resultVec[] = new float[4];

float[] geomagneticValuesAdjusted = new float[4];

geomagneticValuesAdjusted[0] = geomagneticValues[0];

geomagneticValuesAdjusted[1] = geomagneticValues[1];

geomagneticValuesAdjusted[2] = geomagneticValues[2];

geomagneticValuesAdjusted[3] = 0;

Matrix.multiplyMV(resultVec, 0, RsInv, 0, geomagneticValuesAdjusted, 0);

for (int i = 0; i < resultVec.length; i++) {

resultVec[i] = resultVec[i] \* nanoTtoGRate \* gToCountRate;

}

if (kalmanFiletring) {

KalmanState currentKalmanStateX = new KalmanState(MEASUREMENT\_NOISE, accelerometerValues[0], (double)resultVec[0], previousKalmanStateX);

previousKalmanStateX = currentKalmanStateX;

KalmanState currentKalmanStateY = new KalmanState(MEASUREMENT\_NOISE, accelerometerValues[1], (double)resultVec[1], previousKalmanStateY);

previousKalmanStateY = currentKalmanStateY;

KalmanState currentKalmanStateZ = new KalmanState(MEASUREMENT\_NOISE, accelerometerValues[2], (double)resultVec[2], previousKalmanStateZ);

previousKalmanStateZ = currentKalmanStateZ;

if (previousKalmanStateCounter == KALMAN\_STATE\_MAX\_SIZE) {

magneticXTextView.setText("x: " + previousKalmanStateX.getX\_estimate());

magneticYTextView.setText("y: " + previousKalmanStateY.getX\_estimate());

magneticZTextView.setText("z: " + previousKalmanStateZ.getX\_estimate());

resetKalmanFilter();

} else {

previousKalmanStateCounter++;

}

} else {

magneticXTextView.setText("x: " + resultVec[0]);

magneticYTextView.setText("y: " + resultVec[1]);

magneticZTextView.setText("z: " + resultVec[2]);

}

}

}

}

}

private void resetKalmanFilter() {

previousKalmanStateX = null;

previousKalmanStateY = null;

previousKalmanStateZ = null;

previousKalmanStateCounter = 0;

}

@Override

public void onAccuracyChanged(Sensor sensor, int i) {

}

}

En mi comentario sobre la respuesta marcada en el enlace que proporcionó anteriormente, me referí a mi respuesta simple al calcular la aceleración en referencia al norte verdadero  
  
Permítanme responder aquí de nuevo con más aclaraciones. La respuesta es el producto de la matriz de rotación y los valores del campo magnético. Si lees más sobre la "X siempre es muy pequeña" es el valor correcto.  
  
El acelerómetro y los sensores de campo magnético miden la aceleración del dispositivo y el campo magnético de la tierra en la ubicación del dispositivo, respectivamente. Son vectores en 3 espacios dimensionales, llamémoslos a y m respectivamente.  
Si te quedas quieto y giras tu dispositivo, teóricamente m no cambia, suponiendo que no haya interferencia magnética de los objetos circundantes (en realidad m debería cambiar poco, si te mueves porque el campo magnético de la tierra debería cambiar poco en una distancia corta). Pero un cambio sí, a pesar de que no debería ser drástico en la mayoría de las situaciones.  
  
Ahora un vector v en el espacio tridimensional puede representarse con 3 tuplas (v\_1, v\_2, v\_3) con respecto a alguna base (e\_1, e\_2, e\_3), es decir, v = v\_1 e\_1 + v\_2 e\_2 + v\_3 e\_3. (v\_1, v\_2, v\_3) se llaman las coordenadas de v con respecto a la base (e\_1, e\_2, e\_3).  
  
En dispositivos Android, la base es (x, y, z) donde, para la mayoría de los teléfonos, x está en el lado más corto y apunta a la derecha, y está en el lado más largo y apuntando hacia arriba yz es perpendicular a la pantalla y apuntando hacia afuera.  
Ahora esta base cambia a medida que cambia la posición del dispositivo. Uno puede pensar estas bases como una función del tiempo (x (t), y (t), z (t)), en términos matemáticos es un sistema de coordenadas en movimiento.  
  
Por lo tanto, aunque m no cambie, los valores de event.values ​​devueltos por los sensores son diferentes porque la base es diferente (hablaré de la fluctuación más adelante). Como es, los valores del evento son inútiles porque nos da las coordenadas, pero no sabemos cuál es la base, es decir, con respecto a alguna base que conocemos.  
  
Ahora la pregunta es: ¿es posible encontrar las coordenadas de a y m con respecto a la base mundial fija (w\_1, w\_2, w\_3) donde w\_1 apunta hacia el este, w\_2 apunta hacia el norte magnético y w\_3 apunta hacia el cielo?  
  
La respuesta es sí siempre que se cumplan 2 supuestos importantes.  
Con estas 2 suposiciones, es simple calcular (solo unos cuantos productos cruzados) el cambio de la matriz de base R desde la base (x, y, z) hasta la base (w\_1, w\_2, w\_3), que en Android se denomina Rotación matriz. Entonces las coordenadas de un vector v con respecto a la base (w\_1, w\_2, w\_3) se obtienen multiplicando R con las coordenadas de v con respecto a (x, y, z). Por lo tanto, las coordenadas de m con respecto al sistema de coordenadas del mundo son solo el producto de la matriz de rotación y los valores de evento devueltos por el sensor TYPE\_MAGNETIC\_FIELD y de manera similar para a.  
  
En android, la matriz de rotación se obtiene llamando a getRotationMatrix (float [] R, float [] I, float [] gravity, float [] geomagnética) y normalmente pasamos los valores del acelerómetro devueltos para el parámetro de gravedad y los valores del campo magnético para el geomagnético  
  
Las 2 suposiciones importantes son:  
1- El parámetro de gravedad representa un vector que se encuentra en w\_3, más particularmente es el menos del vector influenciado solo por la gravedad.  
Por lo tanto, si pasa los valores del acelerómetro sin filtrar, la matriz de rotación estará un poco apagada. Es por eso que necesita filtrar el acelerómetro para que los valores del filtro sean aproximadamente el menos el vector de gravedad. Dado que la aceleración gravitacional es el factor dominante en el vector acelerómetro, normalmente es suficiente el filtro de paso bajo.  
2- El parámetro geomagnético representa un vector que se encuentra en el plano abarcado por los vectores w\_2 y w\_3. Eso es, yace en el plano del Cielo Norte. Por lo tanto, en términos de la base (w\_1, w\_2, w\_3), la primera coordenada debe ser 0. Por lo tanto, la "X es siempre muy pequeña", como dijiste antes, es el valor correcto, idealmente debería ser 0. Ahora el los valores de campo fluctuarán bastante. Esto es algo esperado, al igual que una aguja de brújula regular no se detendrá si la mantienes en tu mano y tu mano tiembla un poco. Además, puede recibir interferencias de los objetos que lo rodean y, en este caso, los valores del campo magnético son impredecibles. Una vez probé la aplicación de mi brújula sentada cerca de una mesa de "piedra" y mi brújula estaba desconectada más de 90 grados, solo usando una brújula real descubrí que no había nada de malo en mi aplicación y la mesa de "piedra" produce una campo magnético fuerte real.  
Con la gravedad como factor dominante, puede filtrar los valores del acelerómetro, pero sin ningún otro conocimiento, ¿cómo se ajustan los valores magnéticos? ¿Cómo se puede saber si hay o no interferencia de los objetos circundantes?  
  
Puede hacer mucho más como un conocimiento completo de la posición espacial de su dispositivo, etc. con la comprensión de la matriz de rotación.

Según la explicación anterior, haz esto

private static final int TEST\_GRAV = Sensor.TYPE\_ACCELEROMETER;

private static final int TEST\_MAG = Sensor.TYPE\_MAGNETIC\_FIELD;

private final float alpha = (float) 0.8;

private float gravity[] = new float[3];

private float magnetic[] = new float[3];

public void onSensorChanged(SensorEvent event) {

Sensor sensor = event.sensor;

if (sensor.getType() == TEST\_GRAV) {

// Isolate the force of gravity with the low-pass filter.

gravity[0] = alpha \* gravity[0] + (1 - alpha) \* event.values[0];

gravity[1] = alpha \* gravity[1] + (1 - alpha) \* event.values[1];

gravity[2] = alpha \* gravity[2] + (1 - alpha) \* event.values[2];

} else if (sensor.getType() == TEST\_MAG) {

magnetic[0] = event.values[0];

magnetic[1] = event.values[1];

magnetic[2] = event.values[2];

float[] R = new float[9];

float[] I = new float[9];

SensorManager.getRotationMatrix(R, I, gravity, magnetic);

float [] A\_D = event.values.clone();

float [] A\_W = new float[3];

A\_W[0] = R[0] \* A\_D[0] + R[1] \* A\_D[1] + R[2] \* A\_D[2];

A\_W[1] = R[3] \* A\_D[0] + R[4] \* A\_D[1] + R[5] \* A\_D[2];

A\_W[2] = R[6] \* A\_D[0] + R[7] \* A\_D[1] + R[8] \* A\_D[2];

Log.d("Field","\nX :"+A\_W[0]+"\nY :"+A\_W[1]+"\nZ :"+A\_W[2]);

}

}